



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

09/862,828

05/22/2001

Neil W. Taylor

971-128

8874

7590

11/21/2006

MICHAEL T. SANDERSON, ESQ  
KING & SCHICKLI, PLLC  
247 NORTH BROADWAY  
LEXINGTON, KY 40507

EXAMINER

SON, LINH L D

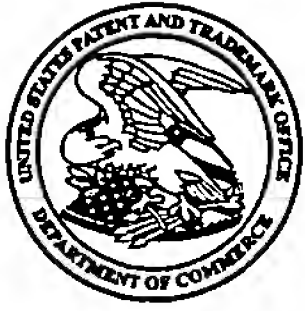
ART UNIT

PAPER NUMBER

2135

DATE MAILED: 11/21/2006

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

---

Commissioner for Patents  
United States Patent and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

**MAILED**

NOV 21 2006

*Technology Center 2100*

**BEFORE THE BOARD OF PATENT APPEALS  
AND INTERFERENCES**

Application Number: 09/862,828  
Filing Date: May 22, 2001  
Appellant(s): TAYLOR, NEIL W.

---

Michael T. Sanderson  
Reg. No. 43,082  
For Appellant

**EXAMINER'S ANSWER**

Art Unit: 2135

This is in response to the appeal brief filed 08/14/2006 appealing from the Office action mailed 04/06/2006.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

The examiner is not aware of any related appeals, interferences, or judicial proceedings, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

The statement of the status of claims contained in the brief is correct.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

5493649	Slivka et al	2-1996
5944821	Angelo	8-1999

**(9) Grounds of Rejection**

a. The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-4, and 7-21 are rejected under 35 U.S.C. 103(a). These rejections are fully set forth in a prior Office action mailed 04/06/2006.

i. Claim s 1-4, and 7-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Slivka et al, US/5493649, hereinafter "Slivka", in view of Angelo, US/5944821, (Cited in PTO 892 dated in 10/03/05 and 05/19/05).

ii. As per claims 1 and 16:

Slivka discloses "A method for validating executable code resident in an operating system having executable instructions, comprising the steps of:

identifying an executable code having an unaltered size (not corrupted)" in (Col 3 lines 1-25, and Col 4 lines 1-5);

Art Unit: 2135

“calculating an initial score associated with the executable code when the executable code is initially or shortly thereafter loaded into an operating system; saving the initial score” in (Col 3 lines 15-43);

calculating a plurality of subsequent score on the executable code; exclusively comparing each of the comparing the subsequent score to the saved initial score and to no other scores” in (See Abstract, Col 3 lines 43-67);

“if the each of the subsequent scores do not vary from the saved initial score, concluding the executable code maintains the unaltered size; and

“if any of the subsequent scores vary from the saved score, concluding the executable code has an altered size (Corrupted program code, which is not equal)” in (Col 3 line 67 to Col 4 line 15). Slivka does disclose a method of calculating a plurality of subsequence score on the executable code after the executable code is launched for use in (Col 5 lines 30-50)

However, Slivka does not specifically disclose the limitation “one of randomly and substantially each time the executable code is launched for use, calculating a plurality of subsequent scores on the executable code;”

Nevertheless, Angelo does disclose a method of calculating a score substantially each time the executable code is launched for use in (Col 9 line 45 to Col 10 line 35).

Therefore, it would have been obvious at the time of the invention was made for one having ordinary skill in the art to modify Slivka’s invention to incorporate Angelo secure executing the executable code everytime it is being launched or requested with

Art Unit: 2135

a motivation of providing safestart everytime executing a program (Angelo, Col 9 lines 25-35).

iii. As per claims 2, and 13:

Slivka discloses "The method of claims 1 and 8, further comprising the steps of: unloading the executable code from the operating system if the saved score is not equal to the subsequent score" in (Col 5 lines 3-10).

iv. As per claims 3 and 17:

Slivka discloses "The method of claims 1 and 16, further comprising the steps of: disabling at least a portion of the executable code if the saved initial score is not equal to any of the subsequent score" in (Col 5 lines 3-10)

v. As per claims 4 and 10:

Slivka discloses "The method of claim 1, wherein the scores are the result of a checksum calculation" in (Col 3 lines 30-35).

vi. As per claims 7, 9, and 20:

Slivka discloses "The method of claims 1, 8, and 16, further comprising the steps of: notifying electronically an owner of the executable code if the saved initial score is not equal to any of the subsequent score" in (Col 5 lines 3-10).

vii. As per claims 8 and 21-22:

Slivka discloses "A method for disabling executable code which has been modified without authorization having executable instructions, "comprising the steps of: identifying an executable code in an operating system having an unaltered format (not corrupted)" in (Col 3 lines 1-25, and Col 4 lines 1-5); "calculating a score associated with an the executable code exclusively within an operating system of a computing device independent of a system management mode of operation" in (Col 2 line 55 to Col 2 line 5); "calculating subsequent scores, the determining exclusively comparing each of the subsequent scores to the score with no other score

Art Unit: 2135

comparisons occurring; and disabling the executable code if the score is not equal to any of the subsequent scores" in (Col 3 line 67 to Col 4 line 15). Slivka does disclose a method of calculating a plurality of subsequence score on the executable code after the executable code is launched for use in (Col 5 lines 30-50)

However, Slivka does not specifically disclose the limitation "one of randomly and substantially each time the executable code is launched for use, calculating subsequent scores associated with the executable code;"

Nevertheless, Angelo does disclose a method of calculating a score, substantially each time the executable code is launched for use in (Col 9 line 45 to Col 10 line 35).

Therefore, it would have been obvious at the time of the invention was made for one having ordinary skill in the art to modify Slivka's invention to incorporate Angelo secure executing the executable code everytime it is being launched or requested with a motivation of providing safestart everytime executing a program (Angelo, Col 9 lines 25-35).

viii. As per claims 11-12, and 18:

Slivka discloses "The method of claims 8 and 16, further comprising the steps of: receiving one or more additional scores periodically on the executable code and disabling the executable code if any of the subsequent score is not equal" in (Col 3 lines 40-60).

ix. As per claim 14:

Slivka discloses "The method of claim 8, further comprising the steps of: assisting in the loading of the executable code, if not disabled, to a memory of an operating system wherein the executable code resides" in (Col 5 lines 3-10).

x. As per claim 15:

Slivka discloses "The method of claim 8, further comprising the steps of: registering the executable code if not disabled; and recording a history if the executable code is disabled" in (Col 5 lines 3-10).

xi. As per claim 19:

Slivka discloses "The method of claim 16, wherein the subsequent score is received each time the executable code is initiated in the memory for an execution" in (Col 1 lines 55-67).

#### **(10) Response to Argument**

a. **Whether Angelo supplies the missing teaching of Slivka:**

i. Appellant argues that "*Slivka's one-to-one comparison of a first checksum to a second checksum does not have multiple instances of subsequent checksums, nor does it have exclusive comparisons between the multiple instances to the first checksum and to no others*". (1<sup>st</sup> Paragraph on Page 18 of the Appeal Brief)

Examiner respectfully traverses the Appellant's argument. In Col 3 lines 30-59, Slivka discloses a method of identifying an executable code having an unaltered size. The executable code is the code section of the file management component. Upon an initial starting of the file management component, checksum routine calculates checksum or score for the code section of the file management component and save the checksum or score in the memory #306 of



Figure 3 as an initial score or checksum. The initial starting of the file management component into the operating system is also the "*launching for use*" of the file management component code. The file management component is now ready to perform operations requested. During an instance of "*launching for use*" of the file management component code as above, the file management component calculates many subsequent checksums or scores based on the number of requests performed and the subsequent checksums or scores are compared against the initial checksum or score saved in memory 306 of Fig. 3.

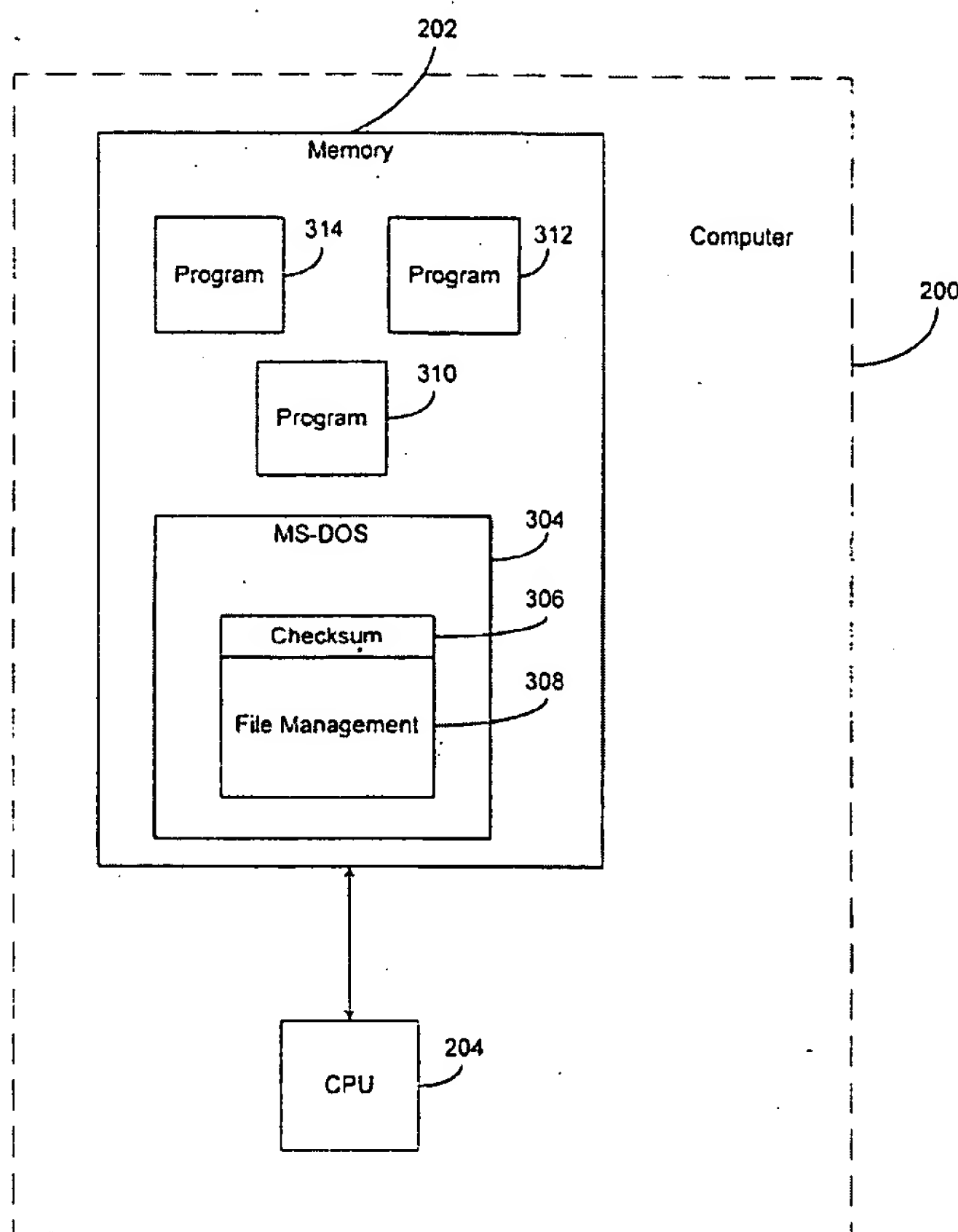


FIG. 3

Art Unit: 2135

However, as disclosed in Slivka, the subsequent scores are only calculated at a time interval or count elapsed. Based on the Appellant's argument on page 20 of the brief, *the time delay interval could cause the checksum comparison from occurring all the time.*

In steps 402 and 404, upon initially starting the file management component (e.g., system startup), the file management component calculates checksums for the code section and the data section of the file management component by invoking the calculate checksum routine. In step 406, the file management component waits until it receives an operation to perform. The file management component, when performing an operation, always executes code as well as either reading from or writing to the data section of the file management component. Since modifications to the code section are very rare, the checksum on the code section of the file management component is preferably not checked upon receipt of every operation request. Therefore, in step 408, the file management component determines whether a periodic interval has elapsed. The periodic interval can be based on timing, count of operations requested, or other suitable events. If the periodic interval has not elapsed, the file management component continues to step 416. However, if the periodic interval has elapsed, in step 410, the file management component calculates a new checksum for the code section by invoking the calculate checksum routine. In step 412, the file management component determines whether both checksums of the code section of the file management component are equivalent. If the checksums are not equivalent, in step 428 the file management component indicates to the computer user that the code section of the file management component has been corrupted and processing ends, before the code is executed. If the two checksums are equivalent, however, processing continues.

Nevertheless, Angelo discloses a method of calculating a checksum or score substantially each time the code or executable is being performed or executed. The calculated checksum or score is compared with a saved checksum or score in the memory in (Angelo, Col 10 lines 15-28)

(28) As part of step 308, the SMI handler 202 next determines if the hash table 206 contains a hash value corresponding to the program to be executed. Typically, a secure hash value is created for each program to be tracked as part of the program's installation into the computer system S. If a hash value for the program is found, control proceeds to step 310 where the stored hash value is retrieved. Control then proceeds to step 312 for a comparison of the newly generated hash value with the

Art Unit: 2135

**stored hash value.** If the two values are the same, control passes to step 318 and the program is loaded into memory and executed. As mentioned, the program or portions of it can be loaded into SMM memory 200 for execution.

Therefore, it would have been obvious at the time of the invention was made for one having ordinary skill in the art to modify Slivka's invention to incorporate Angelo secure executing the executable code every time it is being launched with a motivation of providing safestart every time executing a program (Angelo, Col 9 lines 25-35, and lines 42-45)

(24) In **prior virus and integrity checking systems such as that disclosed in the SAFESTART patent, a secure hash value for the first code to be loaded is stored in NVRAM that is locked down after startup.** By storing this value in SMM memory 200, however, it can be accessed or modified in real time via the secured SMM path. The invention has the additional advantage that extra hardware is not required to secure the NVRAM as was the case in prior systems.

**Of importance to the invention is that each piece of software to be tracked has a corresponding and fairly unique value that represents the unaltered state of the software, and that this value be stored in a secure memory location.**

b. **Whether Slivka's excessive delay in checksum comparisons is remedied by Angelo's program hash routines during safestart modes of operation prior to loading of programs into memory;**

i. As per argument on page 19 continuing to first paragraph on page 20, the Appellant's argument is moot. As clarified above, Examiner does not rely on the checksum calculation of the data component of the file management component.

**Claims 11, 12, and 19:**

ii. As per argument on page 21 last paragraph continuing to second paragraph on page 22, the Appellant argues that Slivka does not disclose, "the subsequent scores are calculated randomly and substantially each time the

Art Unit: 2135

executable code is launched for used", "the subsequent scores are further calculated at one or more predetermined time intervals, and "the subsequent scores are received each time the executable code is initiated in the memory for an execution". These limitations are recited in claims 11, 12, and 19.

Examiner respectfully traverses the Appellant's argument. Similarly to the Examiner traversal above in (Bullet a. i.) is applicable. The initial starting of the file management component into the operating system is the "launching for use" or "the initiating in the memory for an execution" of the file management component code. A subsequent checksums or scores get calculated based on the operation requested based on a predetermined time interval. The file management component calculates a new checksum for comparison if a predetermined time interval is elapsed. That means the checksum is only calculated at random, not every time the code is operated (Col 3 lines 43-50).

c. **Whether Angelo discloses scores or calculations of scores associated with executable code at a time when the code is "initially or shortly thereafter loaded into an operating system" or whether Angelo concerns itself with calculating hash values of programs prior to or before the program being loaded into memory and executed;**

i. As per remark in last paragraph on page 22 continuing to the first paragraph on page 23, Appellant argues that Angelo teaches hash value

Art Unit: 2135

generation and comparison at a time prior to or before loading executable code into an operating system.

Examiner respectfully traverses the Appellant's argument. Angelo discloses a method of validating the program code by calculating an initial score when the program gets installed loaded into an operating system (Col 10 lines 18-20). The calculated initial score is stored in a secure hash table for use to compare all subsequent scores calculated every time the program code gets performed an operation. The method of calculating the subsequent scores and comparing with the initial score every time the program code gets performed an operation is obviously full filling the missing teaching in Slivka.

Therefore, it is obvious at the time of the invention was made for one having ordinary skill in the art to modify Slivka's invention to incorporate Angelo secure executing the executable code every time it is being launched with a motivation of providing safestart every time executing a program (Angelo, Col 9 lines 25-35, and lines 42-45).

d. **Whether Slivka and Angelo are properly combined**

i. Applicant's arguments in the last paragraph on page 23 continuing to page 24 have been fully considered but they are not persuasive. Appellant argues that *"Angelo teaches away from Slivka because Slivka clearly operates in memory 202 other than protected memory"*.

Examiner respectfully traverses the Appellant's argument. In Col 10 lines 28-38, Angelo discloses a method of executing the program in a non-secure memory. Therefore, it is evidenced that Slivka and Angelo is combinable.

ii. In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, the motivation to combine is to provide a safestart every time executing a program in (Angelo, Col 9 lines 25-35, and lines 41-45)

iii.

e. **Whether the Examiner has met his *prima facie* burden**

i. In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Art Unit: 2135

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

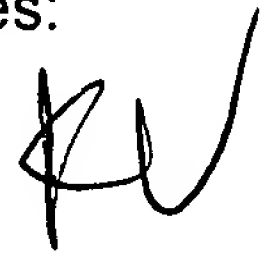
Linh L.D. Son



11/09/2006

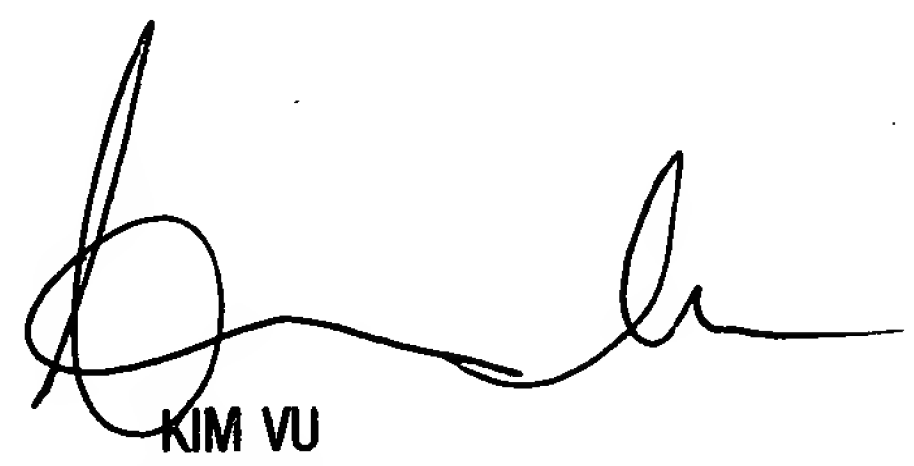
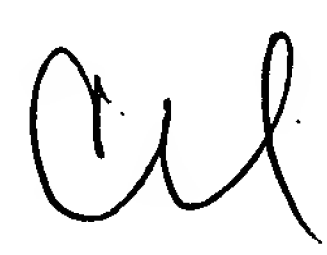
Conferees:

Kim Vu



Christopher Revak

**CHRISTOPHER REVAK  
PRIMARY EXAMINER**



**KIM VU  
SUPERVISORY PATENT EXAMINER  
TECHNOLOGY CENTER 2100**